

C5. Les réseaux convolutionnels

Advanced Machine Learning (MLA)

Kévin Bailly

kevin.bailly@sorbonne-universite.fr

<http://people.isir.upmc.fr/bailly/>



Limites des réseaux MLP

- Comment traiter des images de grande dimension ?

```
class MnistNet(Model):  
    def __init__(self):  
        super(MnistNet, self).__init__()  
        self.flatten = Flatten()  
        self.fc1 = Dense(120, activation='relu')  
        self.fc2 = Dense(80, activation='relu')  
        self.fc3 = Dense(10, activation='softmax')  
  
    def call(self, x):  
        x = self.flatten(x)  
        x = self.fc1(x)  
        x = self.fc2(x)  
        return self.fc3(x)
```

Combien de paramètres
pour la première couche
cachée ?

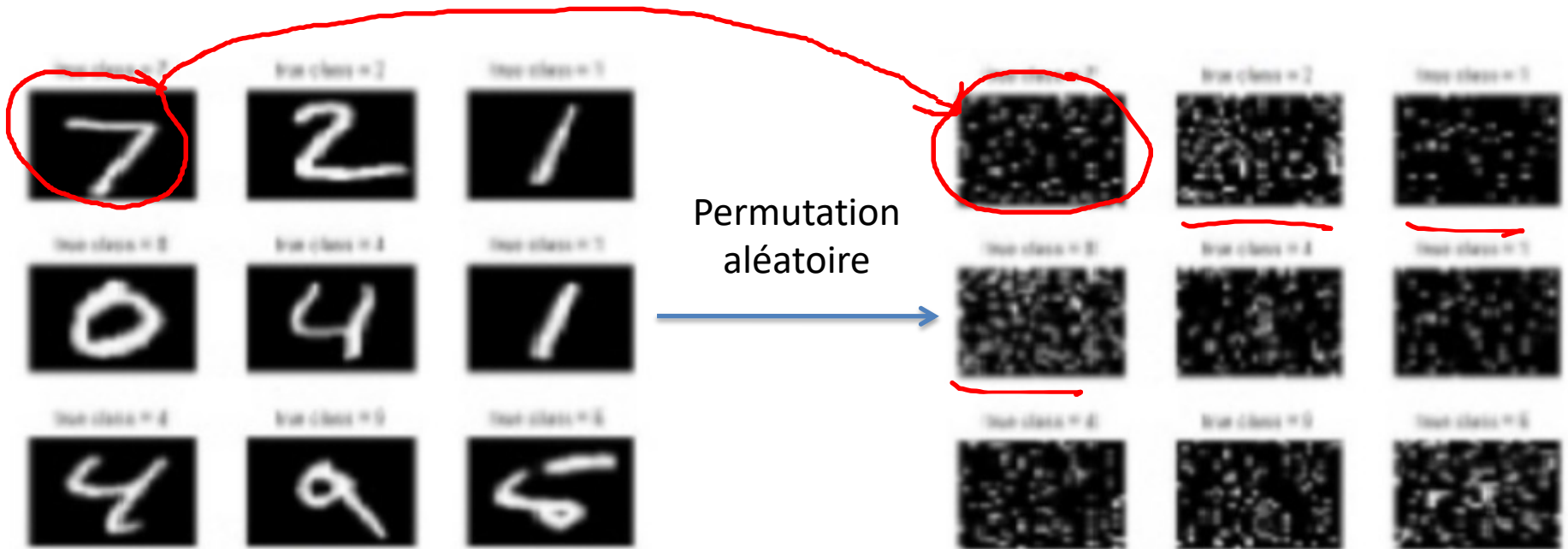
$$28 \times 28 \times 120 = 94080$$

Le nombre de paramètre explose pour des images de plus grande
résolution en RGB - Ex : $640 \times 480 \times 3 \times 120 = 11\text{M} !!$



Limites des réseaux MLP

- Comment conserver la structure spatiale ?



- Un MLP donne-t-il les mêmes résultats sur les deux bases de données ?

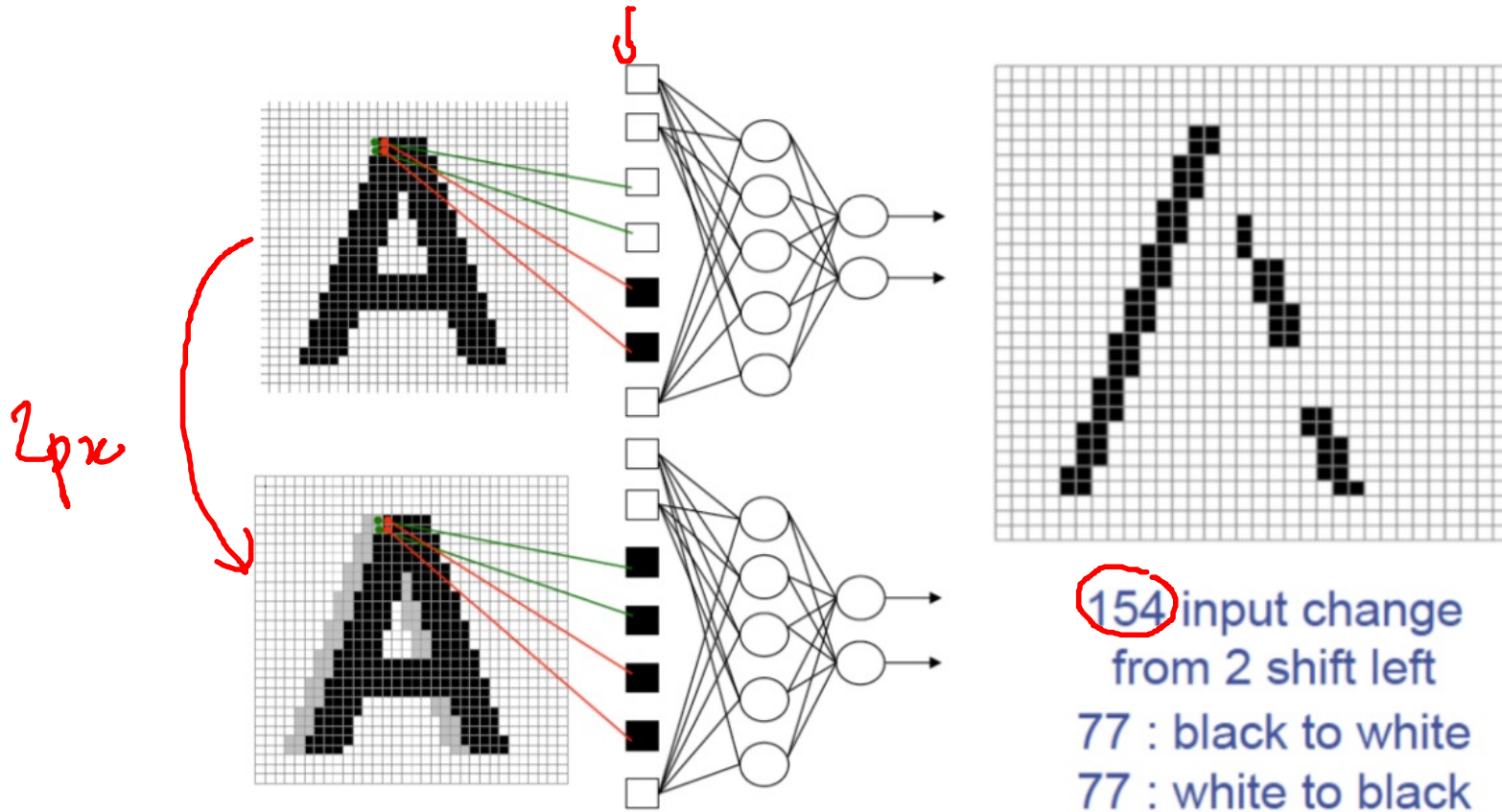
Le codage vectoriel (*flatten*) ne tient pas compte de la structure 2D

→ la structure locale n'est pas prise en compte !



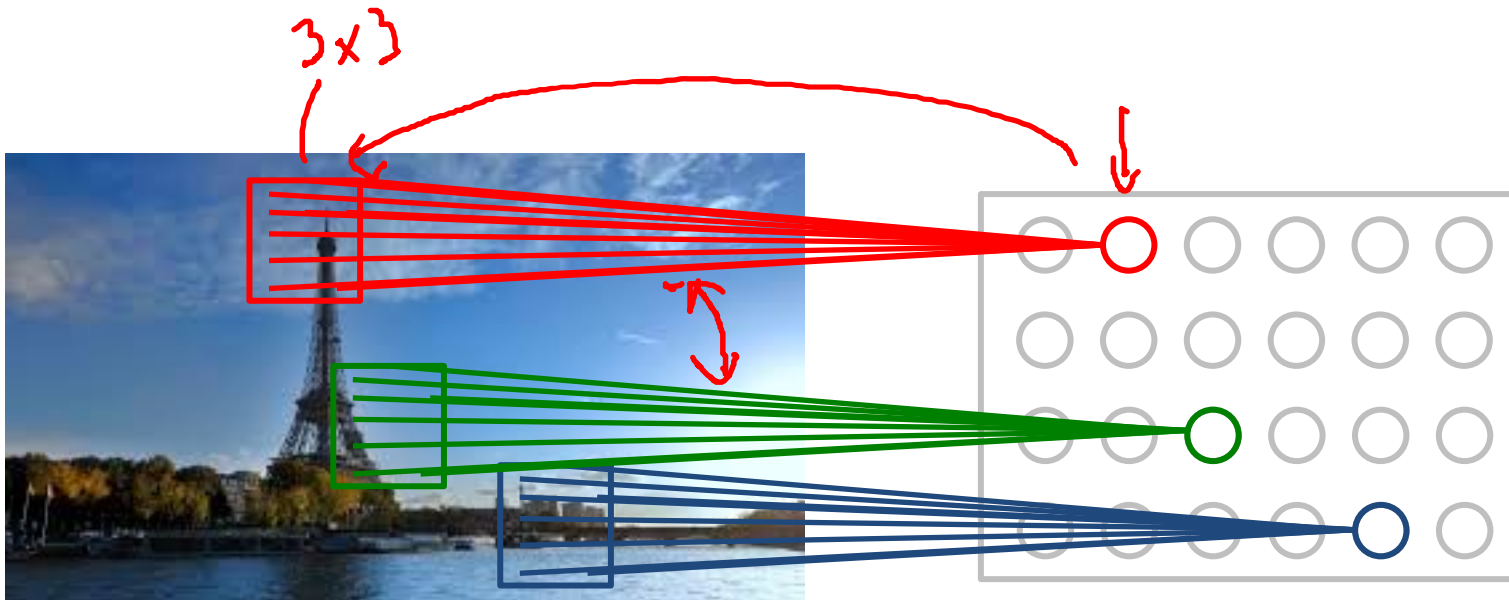
Limites des réseaux MLP

- Comment être moins sensible aux faibles variations



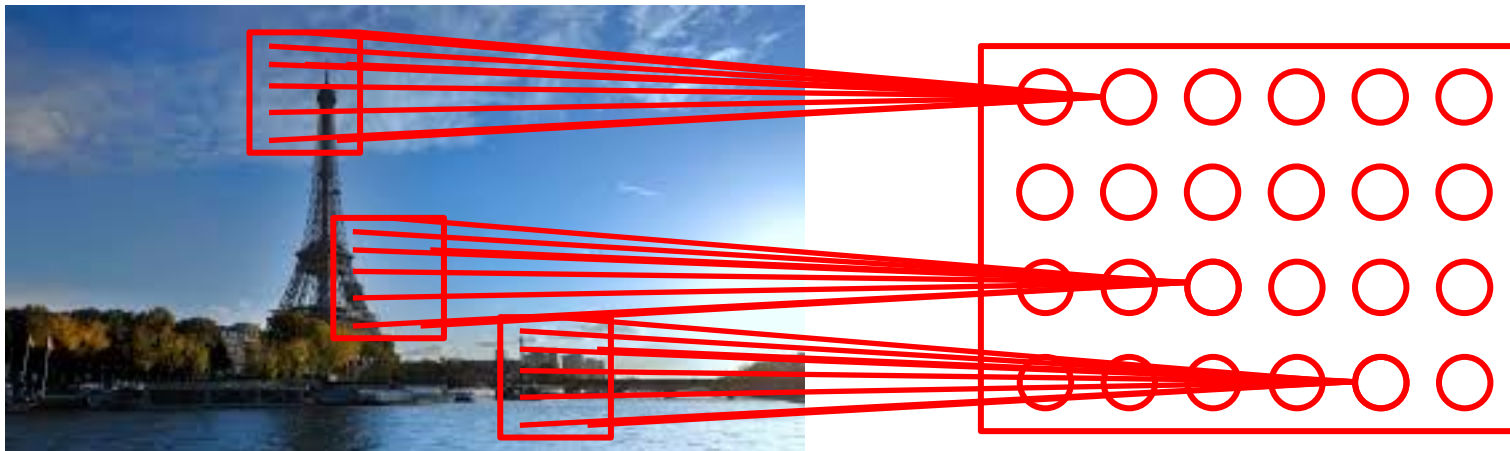
Les réseaux à convolution (CNN)

- Les CNN visent à répondre à ces limitations
 - **Connexions locales** : un neurone est connecté à une sous région (patches locaux)
 - Conservation d'une information spatialement structurée (réponse à des motifs locaux)



Les réseaux à convolution (CNN)

- Les CNN visent à répondre à ces limitations
 - **Connexions locales** : un neurone est connecté à une sous région (patches locaux)
 - **Poids partagés** : même opération répliquée sur tout les neurones
 - Réduction du nombre de paramètres
 - **Convolution avec des filtres appris**

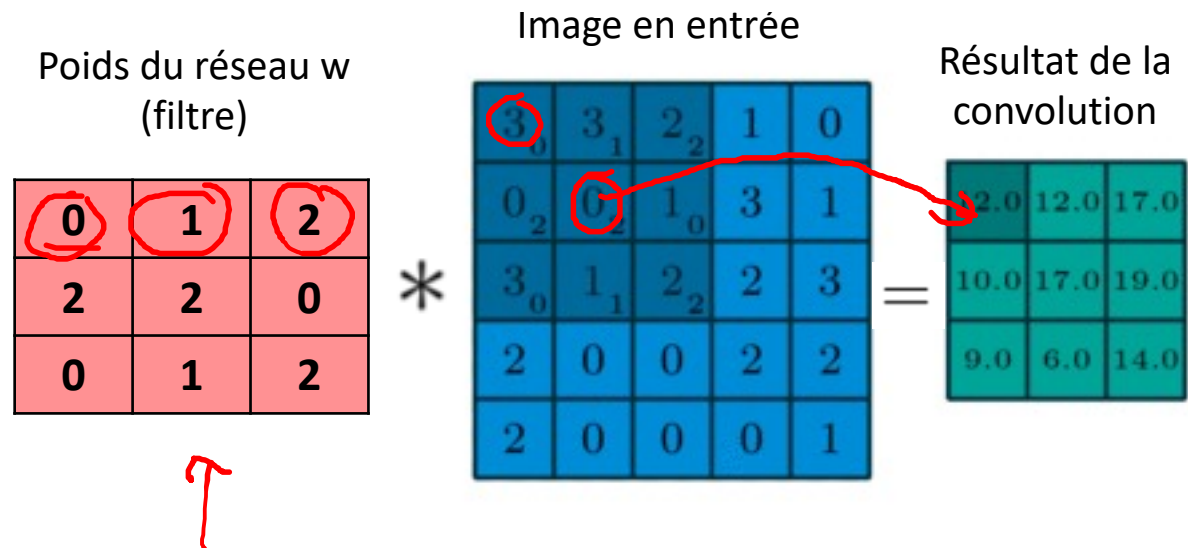


Convolutions

- Chaque neurone (vert foncé) ne « voit » qu'une portion de l'image (bleu foncé)

$$h^n(i, j) = (h^{n-1} * w_k)(i, j) = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} h^{n-1}(i+n, j+m) * w(n, m)$$

Corrélation croisée



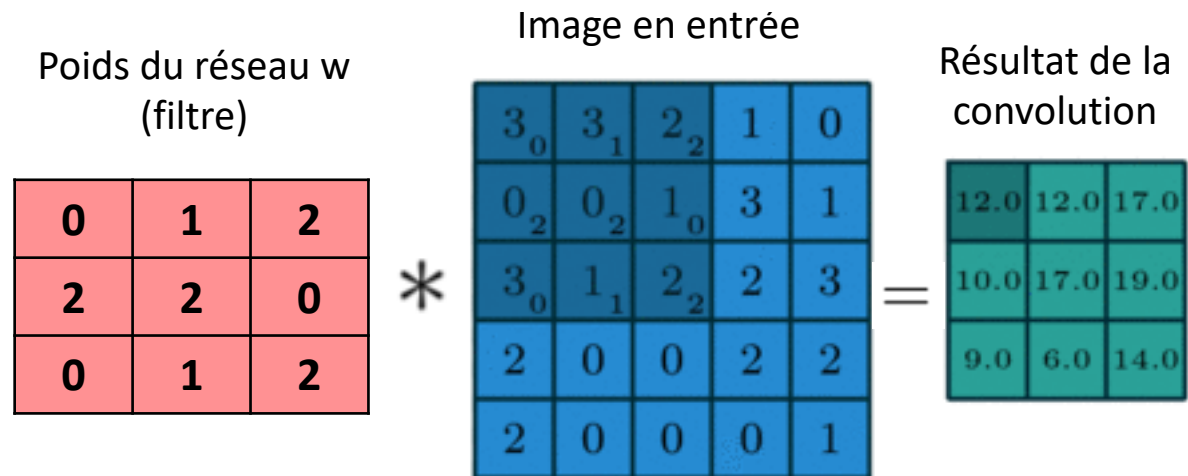
- On applique une non-linéarité (ex. ReLU) sur tous les pixels de la carte d'activation

Convolutions

- Chaque neurone (vert foncé) ne « voit » qu'une portion de l'image (bleu foncé)

$$h^n(i, j) = (h^{n-1} * w_k)(i, j) = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} h^{n-1}(i + n, j + m) * w(n, m)$$

Corrélation croisée

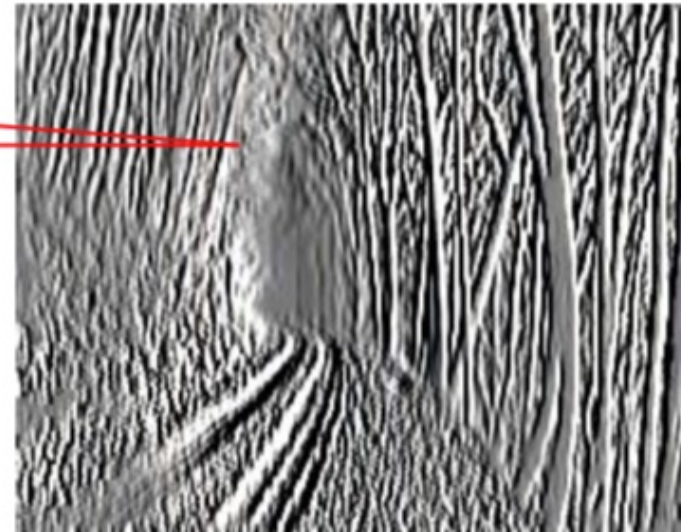


Convolutions



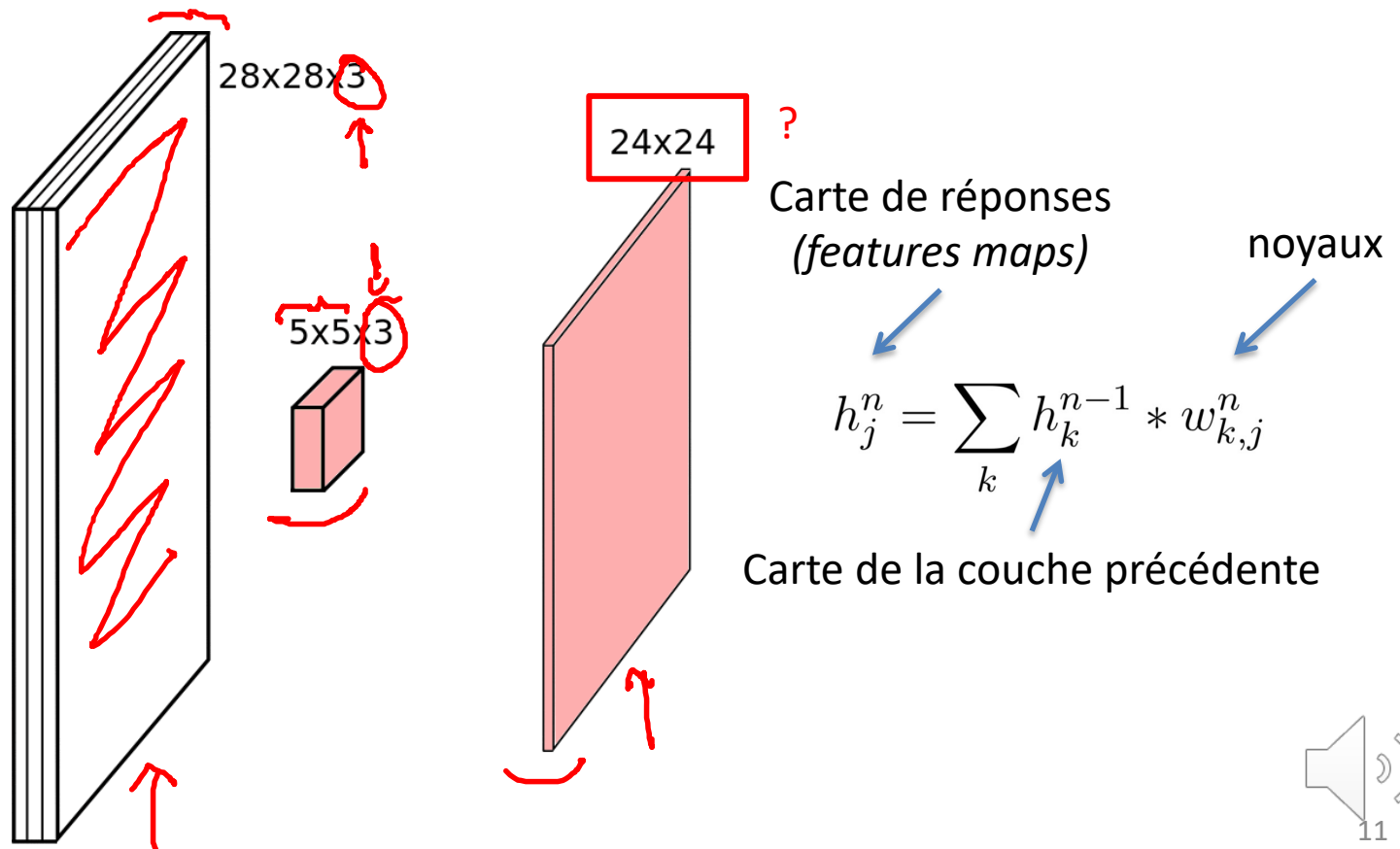
$$\begin{matrix} * \\ \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \\ \end{matrix} =$$

A red curved line is drawn under the 3x3 kernel matrix.

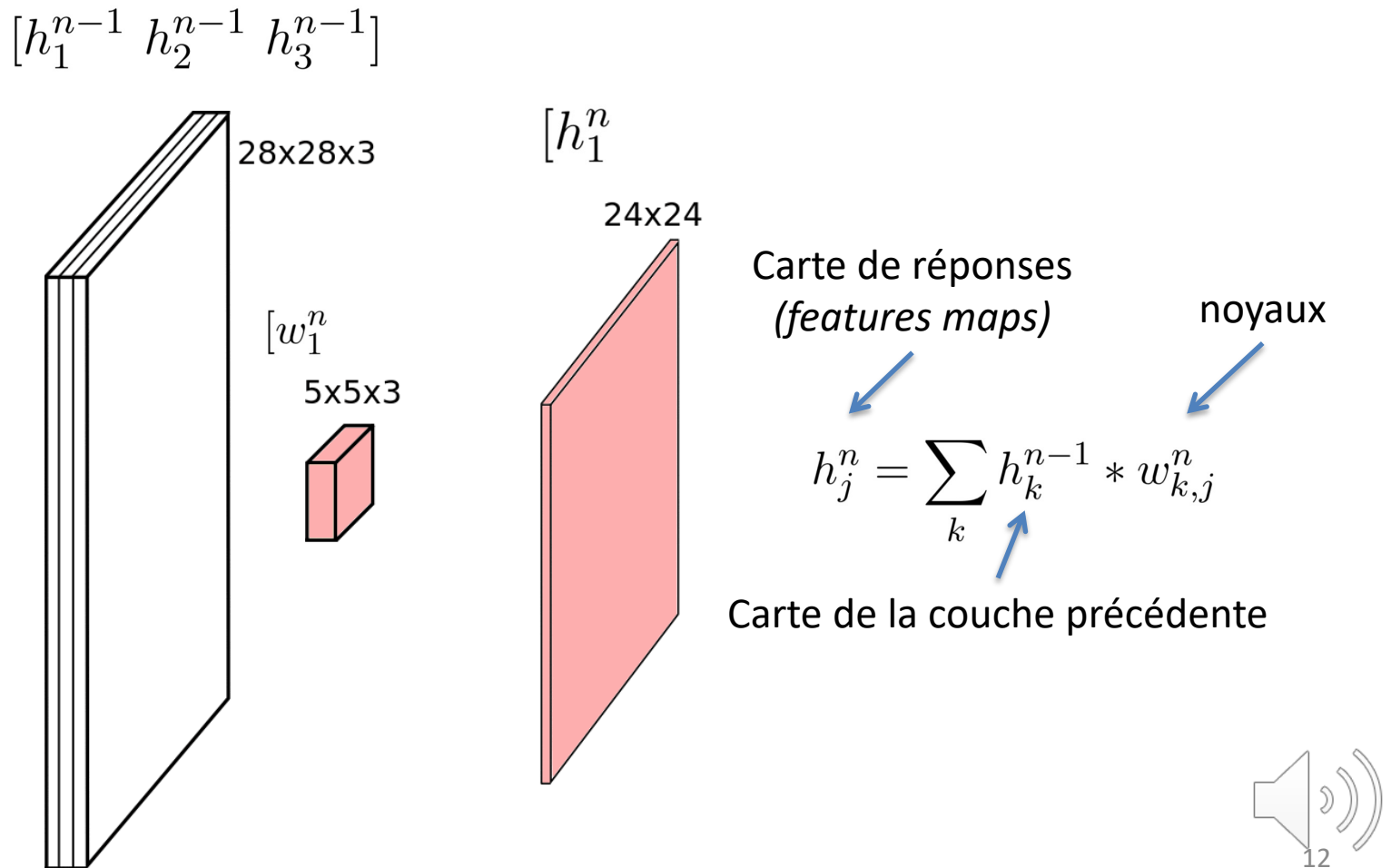


Couches de convolutions

- Une image est un tenseur de taille $[h*w*c]$
- Les convolutions sont calculées pour chaque canal c puis sommées

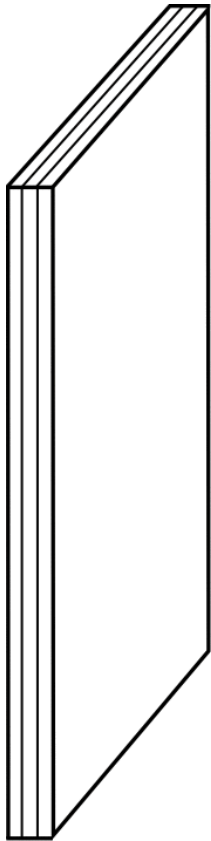


Couches de convolutions



Couches de convolutions

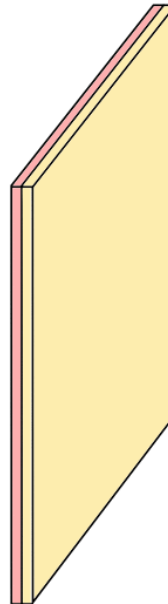
$[h_1^{n-1} \ h_2^{n-1} \ h_3^{n-1}]$



$[w_1^n \ w_2^n]$



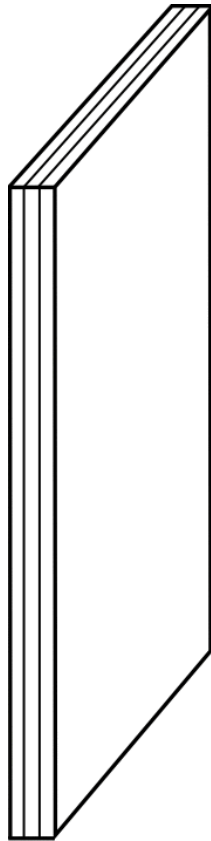
$[h_1^n \ h_2^n]$



$$h_j^n = \sum_k h_k^{n-1} * w_{k,j}^n$$

Couches de convolutions

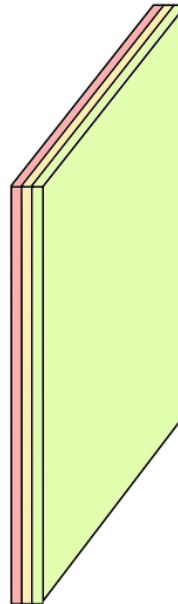
$$[h_1^{n-1} \ h_2^{n-1} \ h_3^{n-1}]$$



$$[w_1^n \ w_2^n \ w_3^n]$$



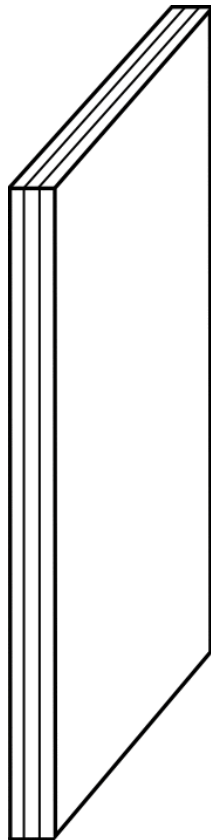
$$[h_1^n \ h_2^n \ h_3^n]$$



$$h_j^n = \sum_k h_k^{n-1} * w_{k,j}^n$$

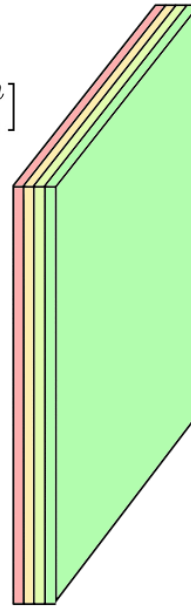
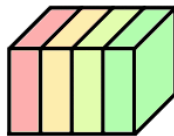
Couches de convolutions

$$[h_1^{n-1} \ h_2^{n-1} \ h_3^{n-1}]$$



$$[h_1^n \ h_2^n \ h_3^n \ h_4^n]$$

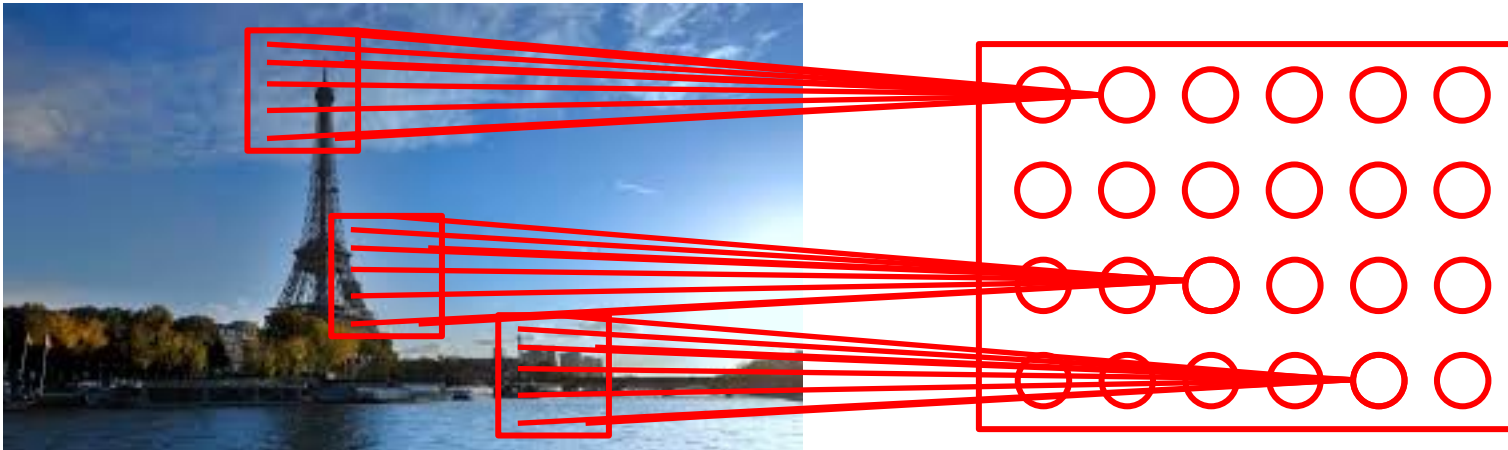
$$[w_1^n \ w_2^n \ w_3^n \ w_4^n]$$



$$h_j^n = \sum_k h_k^{n-1} * w_{k,j}^n$$

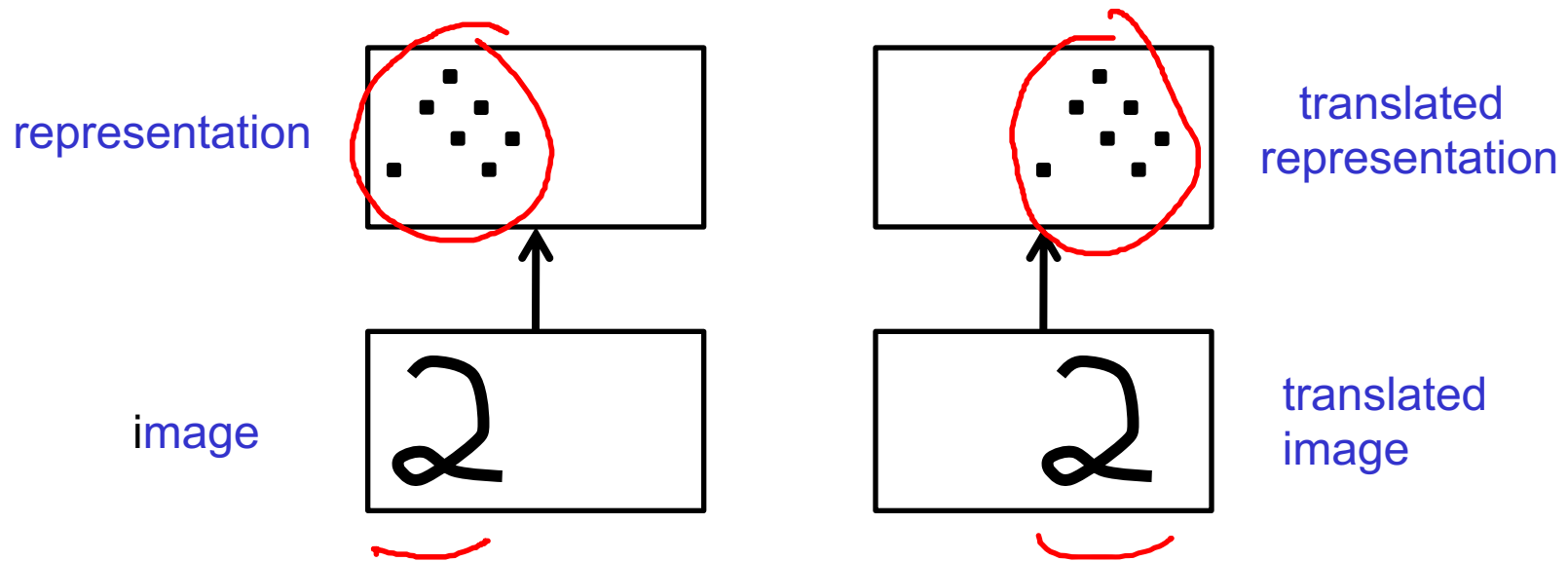
Les réseaux à convolution (CNN)

- Les CNN visent à répondre à ces limitations
 - **Connexions locales** : un neurone est connecté à une sous région (patches locaux)
 - **Poids partagés** : même opération répliquée sur tout les neurones
 - **Et l'invariance aux faibles variations ??**



Equivariance

- Les poids partagés (noyaux) n'assurent **pas une invariance** à la translation mais **une equivariance**

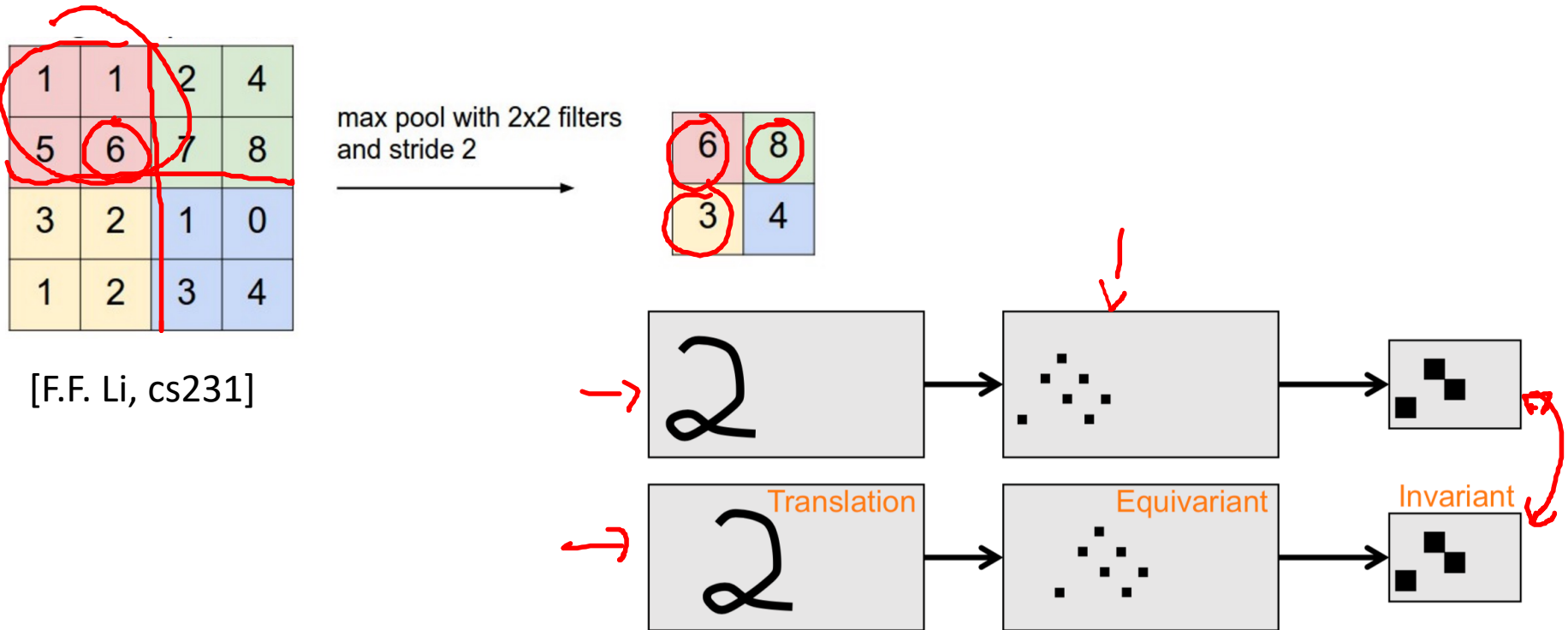


- Comment introduire une invariance aux transformations spatiales 2D ??



L'étape de *Pooling*

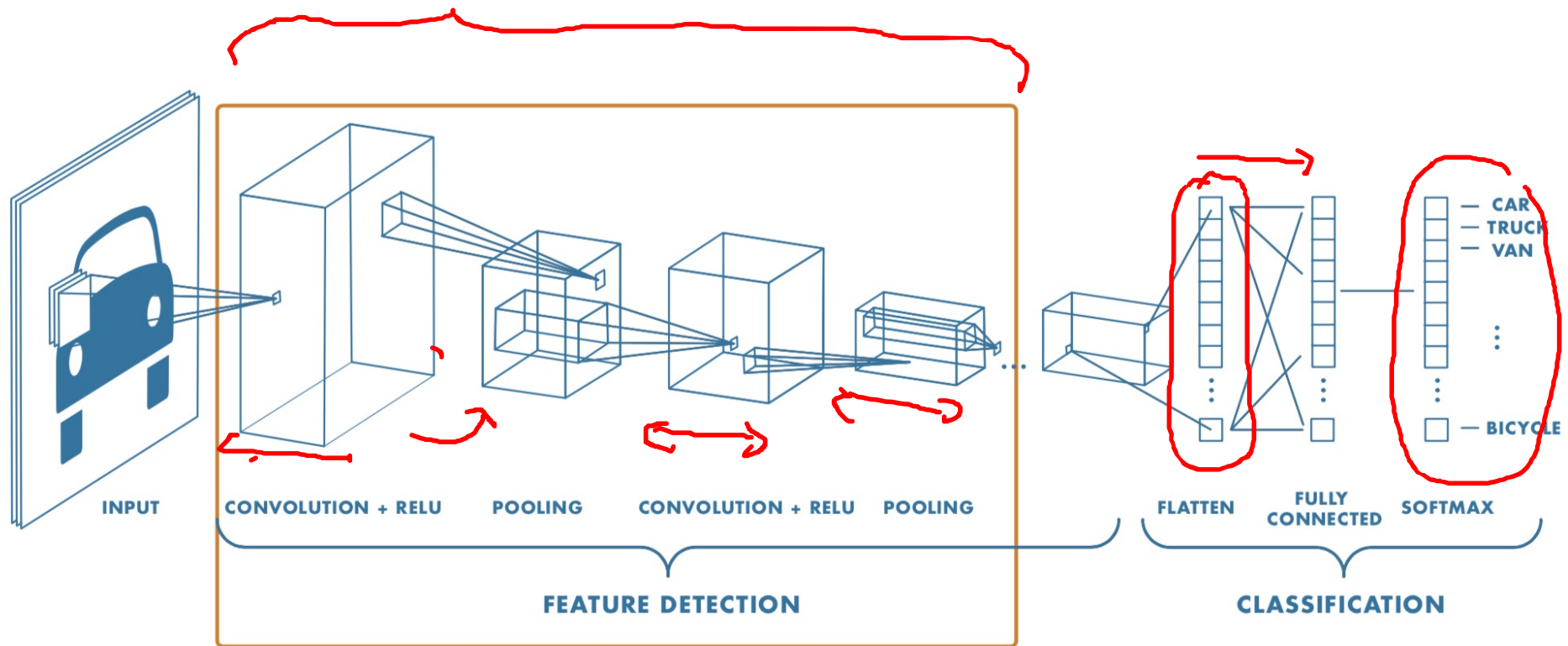
- Un **pooling spatial** (local) des cartes de réponses



- Réduit la dimension des cartes d'activation
- Assure une invariances au faibles translations
- Il existe plusieurs type de pooling (maxpool, meanpool, random...)

Réseau CNN

- Vue globale d'un réseau CNN pour des problèmes de classification

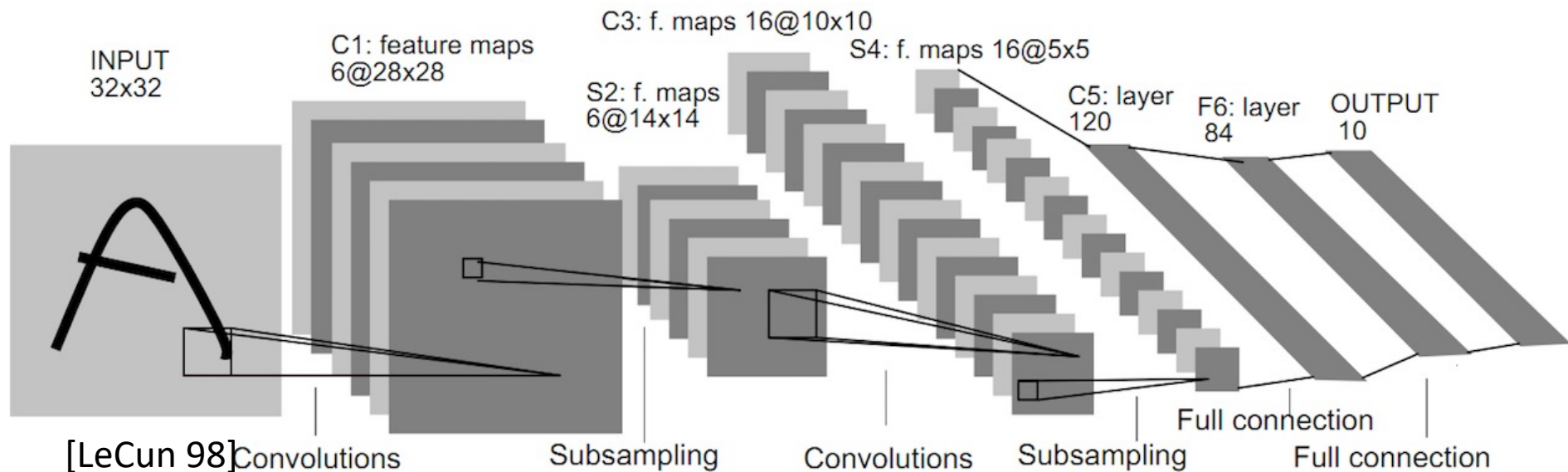


[Mathworks : Introducing Deep Learning with MATLAB]



LeNet-5




- Architecture :
 - 3 couches de convolution C1, C3 et C5 : **taille des filtres ?**
 - 2 couches de « average pooling » : **taille ?**
 - 1 couche complètement connectée (FC, *fully conncteted*) F6
 - Fonction d'activation : tanh

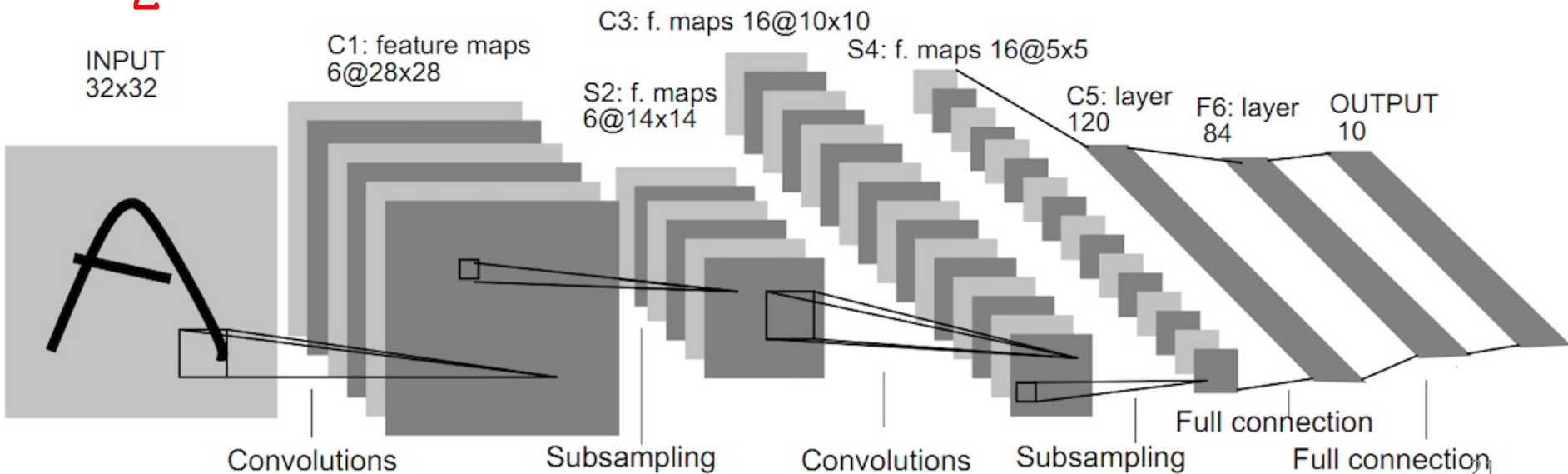


LeNet-5 : spécificités

- 1 couche de sortie « atypique » : 10 fonctions gaussiennes (RBF)

- Couches S2 -> C3 partiellement connectées
→ Réduction du nombre de paramètres
→ Décorrélacion des cartes

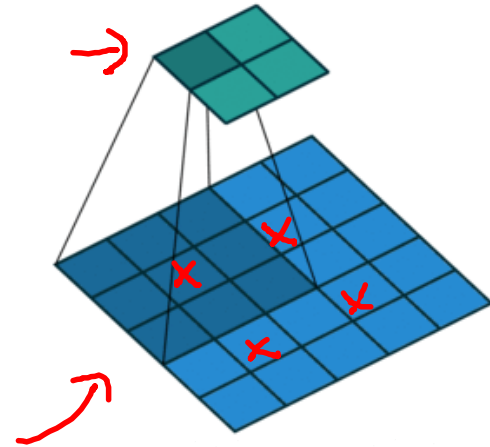
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X



Stride, padding et dilation

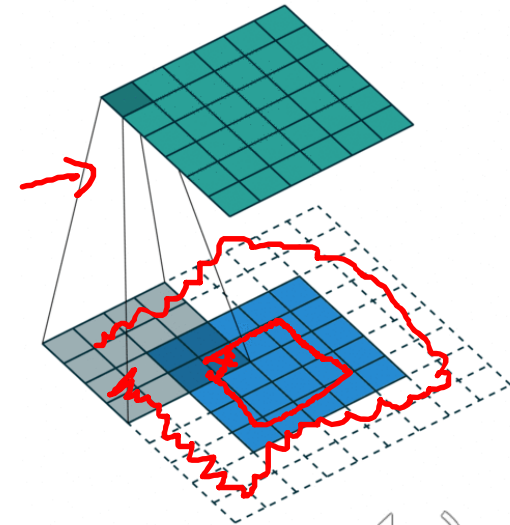
- **Stride**

- Augmente le pas de l'opérateur de convolution
- **Effet ?**
- Réduction de la taille de la carte d'activation



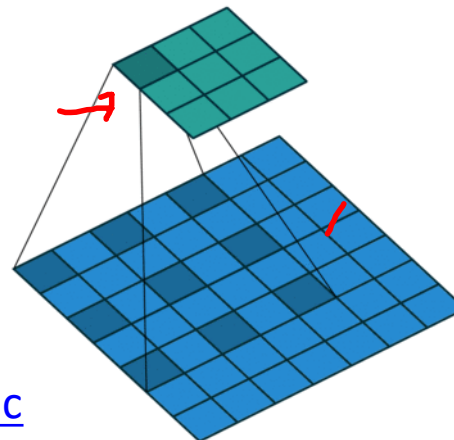
- **Padding**

- Ajoute un bord à l'image (généralement 0)
- Conserve la taille de l'image d'origine



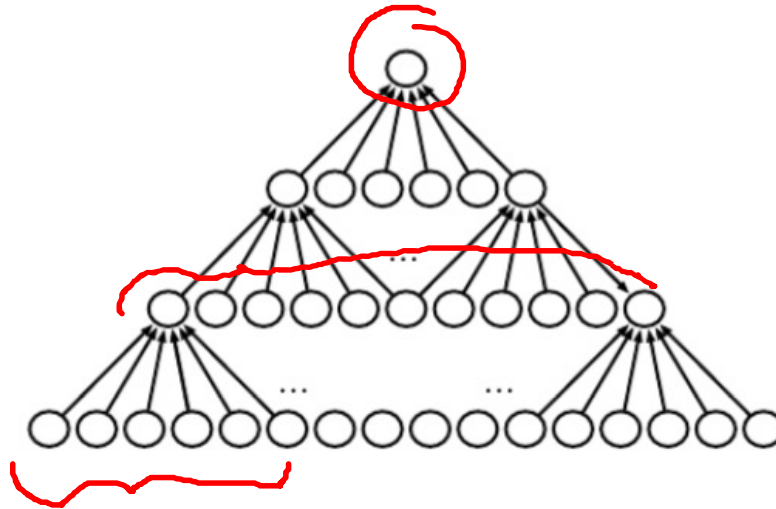
- **Dilation**

- Réduit la taille
- \sim (conv + pool)



Effet du padding

Sans

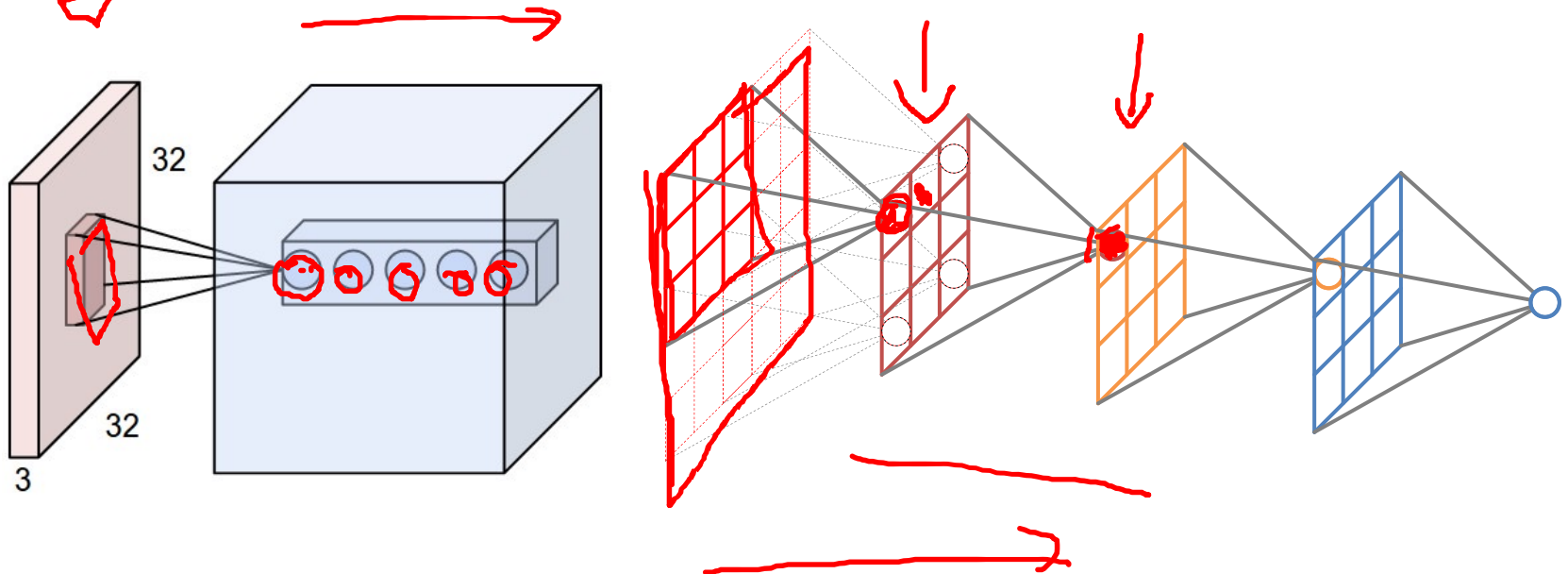


Avec



Champs réceptifs

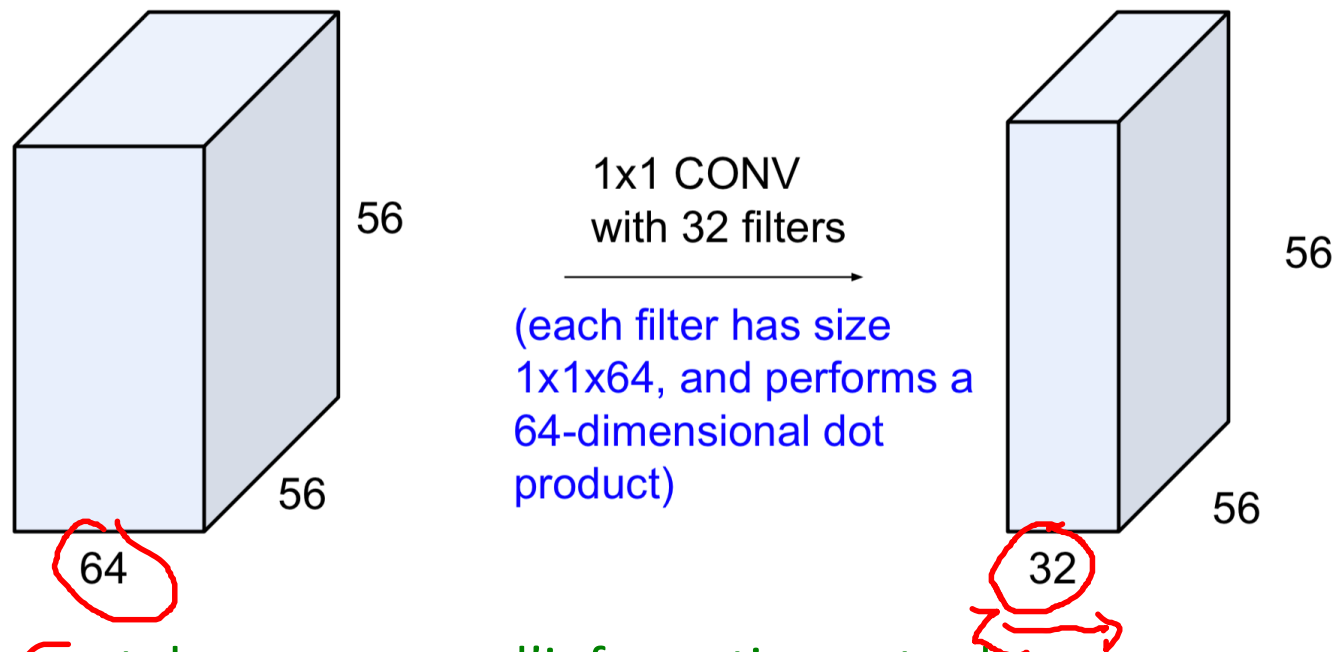
- Portion de l'image qui a un impact sur la décision d'un seul neurone



- Comment évolue le champs réceptif avec la profondeur du réseau ?
 - Il augmente (même sans pooling)
- Est-il préférable d'augmenter la profondeur du réseau ou la largeur des filtres ? Pourquoi ?
 - Augmenter la profondeur limite le nombre de paramètres

Champs réceptifs

- Certain réseaux utilisent des convolution 1x1 **pourquoi ??**

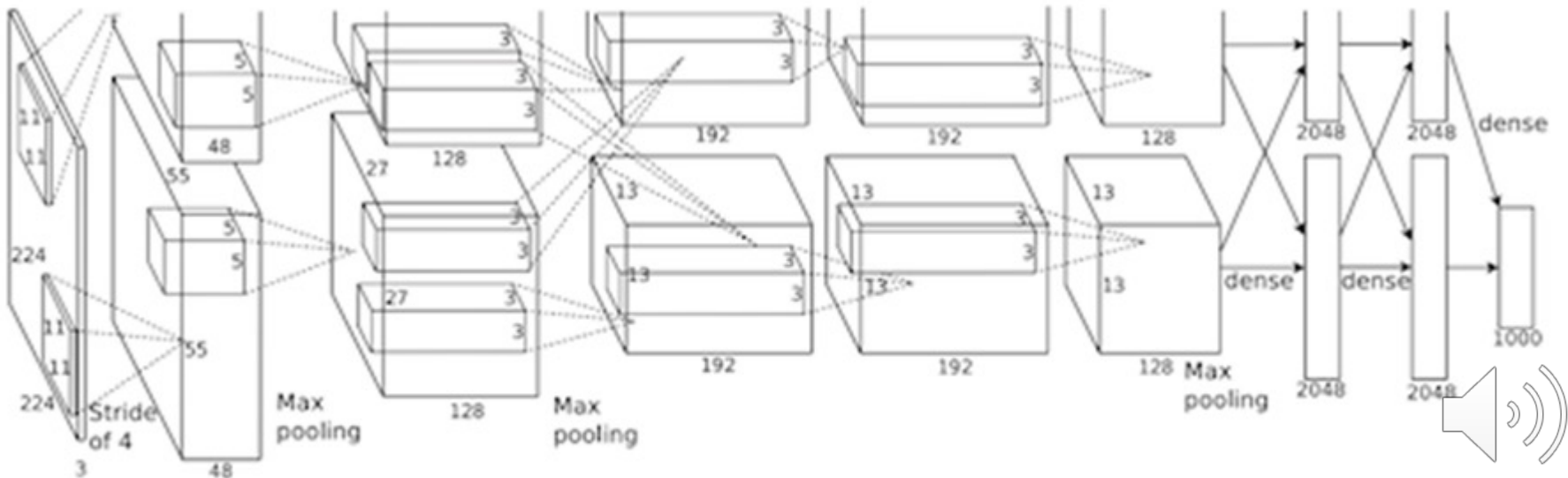


- ~~Permet de compresser l'information entre les canaux~~
- Pour les réseaux très profonds, séparer les convolutions spatiales et inter-canaux permet de limiter le nombre de paramètres (ex: GoogLeNet)

Quelques architectures modernes

- **AlexNet (2012)**

- Premier réseaux CNN à remporter LSVRC (Challenge ImageNet)
 - Top 5 test error rate : 15.4%
- 7 couches
- ReLU (plus rapide à entraîner) + Dropout (meilleure généralisation)
- Augmentation de données (translations réflexions-, patches)
- Gradient stochastique + momentum + weight decay
- 2 architectures en parallèle pour la répartition sur GPU



Quelques architectures modernes

- **VGG Net 2014**

- % Erreur sur ImageNet : 7.3
- 16 couches
- Mais des filtres plus petit
(3x3) au lieu de (11x11) et
(5x5) pour les 1^{ère} c. AlexNet
 - 3 (3x3) \simeq 1(7,7)
- Deep and simple !!

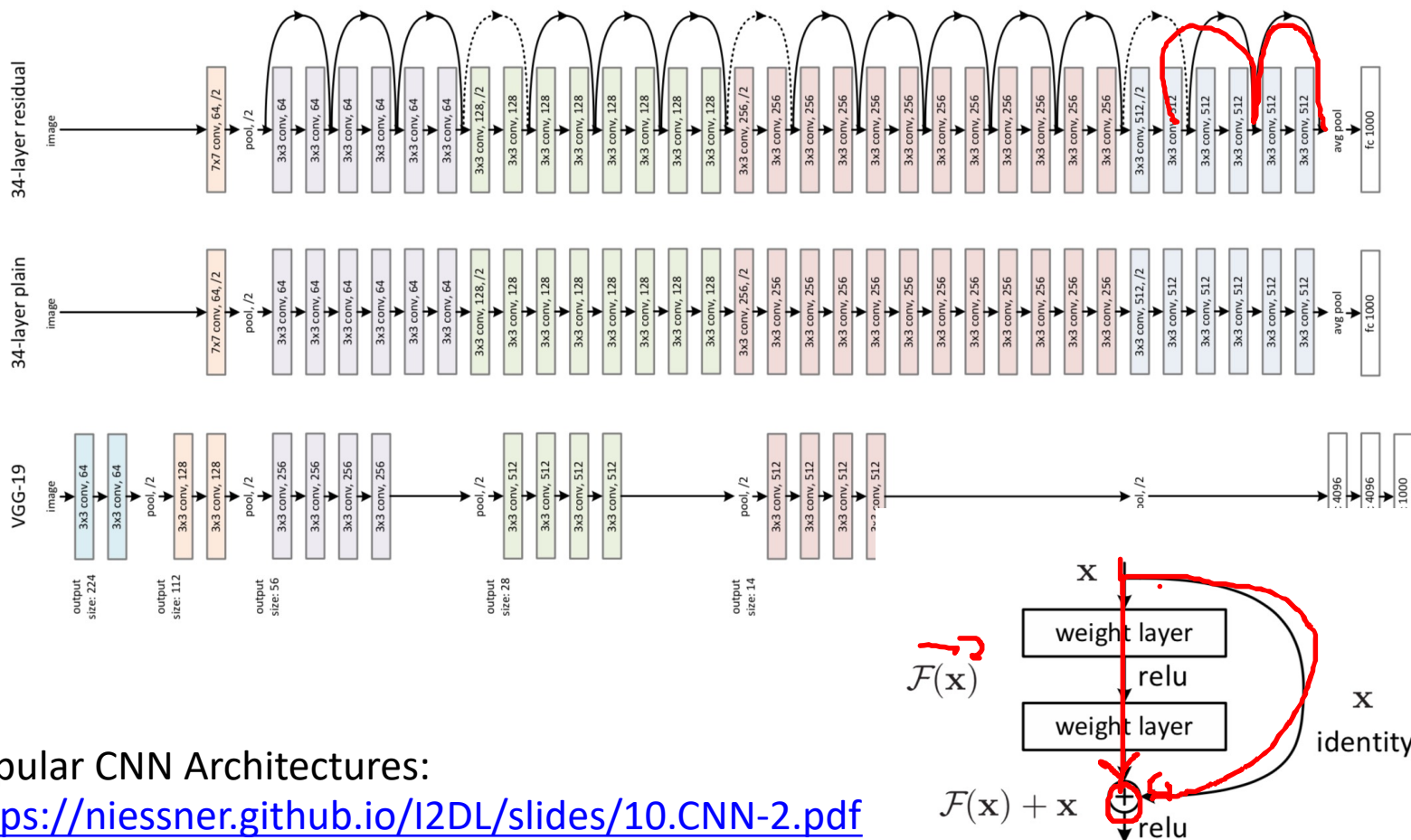
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64	conv3-64	conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128	conv3-128	conv3-128
maxpool					
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256	conv3-256 conv1-256	conv3-256 conv3-256	conv3-256
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512 conv1-512	conv3-512 conv3-512	conv3-512
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512 conv1-512	conv3-512 conv3-512	conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					



Quelques architectures modernes

- Microsoft ResNet (2015)

- 152 couches avec



Popular CNN Architectures:

<https://niessner.github.io/I2DL/slides/10.CNN-2.pdf>